AD-A104 819 MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMA--ETC F/G 9/2
A MODEL FOR UNCOORDINATED DISTRIBUTED COMPUTATION OF FIXED POIN--ETC(U)
SEP 81 D P BERTSEKAS N00014-75-C-1183
UNCLASSIFIED LIDS-P-1144 NL

AD A104819

DTIC FILE COPY

REPORT DOCUMENTATION PAGE

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
	AD-A104 819
4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED
A Model for Uncoordinated Distributed Computation of Fixed Points.	Technical
	6. PERFORMING ORG. REPORT NUMBER: LIDS-P-1144
7. AUTHOR(s): Dimitri P. Bertsekas	8. CONTRACT OR GRANT NUMBER(s): ~~ONR~~ N00014-75-C-1183
9. PERFORMING ORGANIZATION NAME AND ADDRESS: MIT, Laboratory for Information and Decision Systems, Cambridge, MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Code No. 5T10, ONR Identifying No. 049-383
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, Virginia 22209	12. REPORT DATE: September, 1981
	13. NUMBER OF PAGES: 5
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research, Information Systems Program, Code 437, Arlington, Virginia 22217	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

LEVEL II

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We present an algorithmic model for distributed computation of fixed points whereby several processors participate simultaneously in the calculations while while exchanging information via communication links. We place essentially no assumptions on the ordering of computation and communication between processors thereby allowing for completely uncoordinated execution. We find that even under these potentially chaotic circumstances it is possible to solve several important classes of problems including the calculation of fixed points of contraction and monotone mappings arising in linear and nonlinear systems of equations, shortest

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

path problems, and dynamic programming.

September 1981 LIDS-P-1144

A MODEL FOR UNCOORDINATED DISTRIBUTED COMPUTATION OF FIXED POINTS*

DIMITRI P. BERTSEKAS
Laboratory for Information and Decision Systems
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

We present an algorithmic model for distributed computation of fixed points whereby several processors participate simultaneously in the calculations while exchanging information via communication links. We place essentially no assumptions on the ordering of computation and communication between processors thereby allowing for completely uncoordinated execution. We find that even under these potentially chaotic circumstances it is possible to solve several important classes of problems including the calculation of fixed points of contraction and monotone mappings arising in linear and nonlinear systems of equations, shortest path problems, and dynamic programming.

1. INTRODUCTION

There is presently a great deal of interest in distributed implementations of various iterative algorithms whereby the computational load is shared by several processors while coordination is maintained by information exchange via communication links. In most of the work done in this area the starting point is some iterative algorithm which is guaranteed to converge to the correct solution under the usual circumstances of centralized computation in a single processor. The computational load of the typical iteration is then divided in some way between the available processors, and it is assumed that the processors exchange all necessary information regarding the outcomes of the current iteration before a new iteration can begin.

The mode of operation described above may be termed synchronous in the sense that each processor must complete its assigned portion of an iteration and communicate the results to every other processor before a new iteration can begin. This assumption certainly enhances the orderly operation of the algorithm and greatly simplifies the convergence analysis. On the other hand synchronous distributed algorithms also have some obvious implementation disadvantages such as the need for an algorithm initiation and iteration synchronization protocol. Furthermore the speed of computation is limited to that of the slowest processor. It is thus interesting to consider algorithms that can tolerate a more flexible ordering of computation and communication between processors. Such algorithms have so far found applications in computer communication networks like the ARPANET [1] where processor failures are common and it is quite complicated to maintain synchronization between the nodes of the entire network as they execute real-time network functions such as the routing algorithm.

Processor network environments for which weakly coordinated distributed computation seems particularly advantageous typically possess one or more

---

*This research was conducted at the M.I.T. Laboratory for Information and Decision Systems with partial support provided by the Defense Advanced Projects Agency under Contract No. ONR-N00014-75-C-1183.

To be presented at the 19th Annual Allerton Conference on Communication, Control & Computing, Sept. 30-Oct. 2, 1981, Urbana, Illinois

of the following characteristics all of which involve occurance of some type of unpredictable event.

1) Computation nodes and communication links are subject to frequent and/or unexpected failures. (For example packet radio networks).

2) Computation nodes have different and/or time varying speeds of execution. (For example each processor is assigned to a perhaps time varying number of tasks involving computation loads which are not fixed a priori).

3) Computation at various nodes is event driven. (For example in data collection or sensor networks where the timing, and ordering of measurements may not be predictable).

It is possible to consider various degrees of coordination in different types of distributed algorithms. An interesting question is to determine the minimum degree of coordination needed in a given algorithm in order to obtain the correct solution. To this end we consider an extreme model of uncoordinated distributed algorithms whereby computation and communication are performed at each processor completely independently of the progress in other processors. It is perhaps surprising that even under these chaotic circumstances it is still possible to solve correctly important classes of fixed point problems. The complete analysis is given in Ref. [2] for broad classes of dynamic programming and in Ref. [3] for more general fixed point problems involving contraction and monotonicity assumptions. In the present paper we describe the algorithmic model and indicate the type of results that can be shown in some generality.

2. A MODEL FOR DISTRIBUTED UNCOORDINATED FIXED POINT ALGORITHMS

The fixed point problem considered in this paper is defined in terms of a set X, a class F of functions mapping X into the extended real line $[-\infty,+\infty]$, and a mapping T which maps F into itself. We wish to find an element J^* of F such that

$$J^* = T(J^*). \tag{1}$$

or equivalently

$$J^*(x) = T(J^*)(x), \qquad \forall\, x\in X, \tag{2}$$

where $J^*(x)$ and $T(J^*)(x)$ denote the values of the functions J^* and $T(J^*)$ respectively at the typical element $x\in X$. We will assume throughout that T has a unique fixed point J^* within the set F.

We provide some examples:

Example 1: (Fixed points of mappings on R^n). Let X be the finite set

$$X = \{1,2,\ldots,n\},$$

and F be the set of all real-valued functions on X. Then F can be identified with the n-dimensional space R^n in the sense that with each $J\in F$ we can associate the n-dimensional vector $J(1),J(2),\ldots,J(n)$. Similarly T(J) can be identified with the n-dimensional vector $T(J)(1),\ldots,T(J)(n)$, so the fixed point problem (1) amounts to solving the system of n equations

$$J^* = T(J^*) \quad \text{or} \quad J^*(i) = T(J^*)(i), \qquad \forall\, i = 1,\ldots,n \tag{3}$$

with the n unknowns $J^*(1),\ldots,J^*(n)$. It is also evident that any system of n (possibly nonlinear) equations with n unknowns can be formulated into a fixed point problem such as (3).

Example 2: (Shortest path problems). Let (N,L) be a directed graph where $N = \{1,2,\ldots,n\}$ denotes the set of nodes and L denotes the set of links. Let $N(i)$ denote the downstream neighbors of node i, i.e., the set of nodes j for which (i,j) is a link. Assume that each link (i,j) is assigned a positive scalar a_{ij} referred to as its length. Assume also that there is a directed path to node 1 from every other node. Then it is known ([2], p.67) that the shortest path distances $J^*(i)$ to node 1 from all other nodes i solve uniquely the equations

$$J^*(i) = \min_{j\in N(i)} \{a_{ij} + J^*(j)\}, \qquad i \neq 1 \tag{4a}$$

$$J^*(1) = 0 \tag{4b}$$

If we make the identifications $X = \{1,2,\ldots,n\}$, F: Set of all functions mapping X into $[0,+\infty]$, and define $T(J)$ for all $J\in F$ by means of

$$T(J)(i) = \begin{cases} \min_{j\in N(i)} \{a_{ij} + J(j)\} & \text{if } i \neq 1 \\ 0 & \text{if } i = 1 \end{cases} \tag{5}$$

then we find that the fixed point problem (2) reduces to the shortest path problem.

The shortest path problem above is representative of a broad class of dynamic programming problems which can be viewed as special cases of the fixed point problem (2) and can be correctly solved by using the distributed algorithms of this paper (see [3]).

Our algorithmic model can be described in terms of a collection of n computation centers (or processors) referred to as nodes and denoted $1,2,\ldots,n$. The set X is partitioned into n disjoint sets denoted $X_1,\ldots,X_n$, i.e.

$$X = \bigcup_{i=1}^{n} X_i, \qquad X_i \cap X_j = \emptyset, \quad \text{if } i \neq j.$$

Each node i is assigned the responsibility of computing the values of the solution function J^* [c.f. (1),(2)] at all $x\in X_i$.

At each time instant, node i can be in one of three possible states compute, transmit, or idle. In the compute state node i computes a new estimate of the values of the solution function J^* for all $x\in X_i$. In the transmit state node i communicates the estimate obtained from the latest computation to one or more nodes j ($j\neq i$). In the idle state node i does nothing related to the solution of the problem. It is assumed that a node can receive a transmission from other nodes simultaneously with computing or transmitting, but this is not a real restriction since, if needed, a time period in a separate receive state can be lumped into a time period in

the idle state.

We assume that computation and transmission for each node takes place in uninterupted time intervals $[t_1,t_2]$ with $t_1 < t_2$, but do not exclude the possibility that a node may be simultaneously transmitting to more than one nodes nor do we assume that the transmission invervals to these nodes have the same origin and/or termination. We also make no assumptions on the length, timing and sequencing of computation and transmission intervals other than the following:

Assumption (A): There exists a positive scalar P such that, for every node i, every time interval of length P contains at least one computation interval for i and at least one transmission interval from i to each node $j \neq i$.

Each node i also has a buffer B_{ij} for each $j \neq i$ where it stores the latest transmission from j, as well as a buffer B_{ii} where it stores its own estimate of values of the solution function for all $x \epsilon X_i$. The contents of each buffer B_{ij} at time t are denoted J_{ij}^t. Thus J_{ij}^t is, for every t, a function from X_j into $[-\infty,\infty]$ and may be viewed as the estimate by node i of the restriction of the solution function J^* on X_j available at time t. The rules according to which the functions J_{ij}^t are updated are as follows:

1) If $[t_1,t_2]$ is a transmission interval from node j to node i the contents $J_{jj}^{t_1}$ of the buffer B_{jj} at time t_1 are transmitted and entered in the buffer B_{ij} at time t_2, i.e.

$$J_{ij}^{t_2} = J_{jj}^{t_1}. \tag{6}$$

2) If $[t_1,t_2]$ is a computation interval for node i the contents of buffer B_{ii} at time t_2 are replaced by the restriction of the function $T(J_1^{t_1})$ on X_i where, for all t, J_i^t is defined by

$$J_i^t(x) = \begin{cases} J_{ii}^t(x) & \text{if } x \epsilon X_i \\ J_{ij}^t(x) & \text{if } x \epsilon X_j, \quad j \neq i \end{cases} \tag{7}$$

In other words we have

$$J_{ii}^{t_2}(x) = T(J_i^{t_1})(x), \qquad \forall\, x \epsilon X_i \tag{8}$$

3) The contents of a buffer B_{ii} can change only at the end of a computation interval for node i. The contents of a buffer B_{ij}, $j \neq i$ can change only at the end of a transmission interval from j to i.

Additional conditions under which there holds

$$\lim_{t\to\infty} J_i^t(x) = J^*(x), \qquad \forall\, x\epsilon X_i,\ i = 1,\ldots,n \tag{9}$$

may be found in [2],[3]. An interesting aspect of results of this type is that they do not require that the initial processor buffer contents be identical and indeed these initial conditions can vary within a broad range. This means that for problems that are being solved continuously in real time it is not necessary to reset the initial conditions and resynchronize the algorithm each time the problem data changes. As a result the potential for tracking slow variations on the solution function is improved and algorithms implementation is considerably simplified.

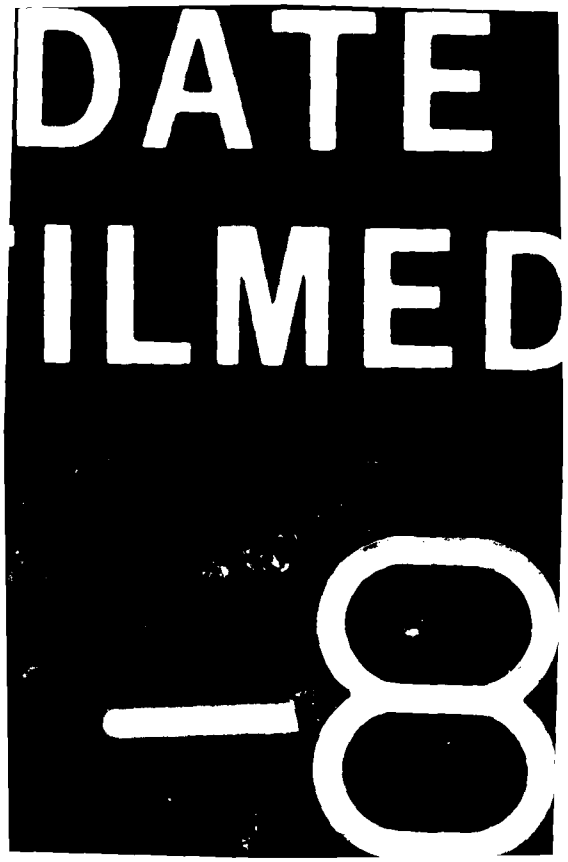
REFERENCES

[1] J. McQuillan, G. Falk, and I. Richer, "A Review of the Development and Performance of the ARPANET Routing Algorithm", IEEE Trans. on Communications, Vol. COM-26, 1978, pp. 1802-1811.

[2] D. P. Bertsekas, "Distributed Dynamic Programming", Report LIDS-P-1060, Laboratory for Information and Decision Systems, Mass. Institute of Technology, Cambridge, Mass., Dec. 1980 (to appear in IEEE Trans. on Automatic Control).

[3] D. P. Bertsekas, "Distributed Computation of Fixed Points", Report LIDS-P-1135, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Mass., Aug. 1981.

[4] E. L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart, and Winston, N.Y., 1976.


DATE
ILMED
8